LOG LINEAR DIALOG MANAGER THAT DETERMINES EXPECTED REWARDS AND USES HIDDEN STATES AND ACTIONS

Applicant: Mitsubishi Electric Research Laboratories, Inc., Cambridge, MA (US)
Inventors: Shinji Watanabe, Arlington, MA (US); Hao Tang, Chicago, IL (US)
Assignee: Mitsubishi Electric Research Laboratories, Inc., Cambridge, MA (US)

Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 305 days.

Appl. No.: 14/106,968
Filed: Dec. 16, 2013

Prior Publication Data

Int. Cl.
G06F 15/18 (2006.01)
G06F 17/30 (2006.01)
G06F 17/27 (2006.01)
G10L 15/22 (2006.01)
G10L 15/18 (2013.01)

U.S. CL.
CPC ........ G06F 17/30976 (2013.01); G06F 17/279 (2013.01); G06F 17/2785 (2013.01); G10L 15/22 (2013.01); G10L 15/1822 (2013.01)

Field of Classification Search
CPC ........ G06N 99/005; G06Q 10/04; G10L 13/00
USPC ................. 706/12, 45

See application file for complete search history.

ABSTRACT
A dialog manager receives previous user actions and previous observations and current observations. Previous and current user states, previous user actions, current user actions, future system actions, and future observations are hypothesized. The user states, the user actions, and the user observations are hidden. A feature vector is extracted based on the user states, the system actions, the user actions, and the observations. An expected reward of each current action is based on a log-linear model using the feature vectors. Then, the current action that has an optimal expected reward is outputted.

13 Claims, 5 Drawing Sheets
Fig. 2

Log-Linear Dialog Trainer (learning part)

\{o_1, ..., o_T\}

{\lambda_{1},...,\lambda_{T}}

\{u_1, ..., u_T\}

\theta_0

\theta_1

\theta_T

\theta_T
Fig. 3
state → route | search | time_to_dest
route → “ROUTE” “[” address “]”
address → “{” “address=” $ADDRESS$ “}”
| “{” “address=” “NULL” “}”
search → “SEARCH” “[” keyword “]”
keyword → “{” “keyword=” $KEYWORD$ “}”
| “{” “keyword=” “NULL” “}”
time_to_dest → “TIME_TO_DEST” “[” route “]”

Fig. 5
LOG LINEAR DIALOG MANAGER THAT DETERMINES EXPECTED REWARDS AND USES HIDDEN STATES AND ACTIONS

FIELD OF THE INVENTION

This invention relates generally to text and speech processing, and more particularly to dialog managers.

BACKGROUND OF THE INVENTION

A dialog manager is a system that accomplishes certain tasks using a dialog, either spoken or text. The dialog alternates between user and system turns. The dialog can include sequences of user actions and system actions. The user actions are hidden from the system. The system determines the user actions from observations. The user has a changing state that is also hidden from the system. The system uses planning to determine a next system action given previous system actions and observations based on user speech or text. The planning is described below.

The dialog manager can be rule based, or use a statistical framework, e.g., a Partially Observable Markov Decision Process (POMDP). In a POMDP dialog system, the dialog is represented by a set of random variables. At each turn, the dialog including an observed variable representing what the user said, a hidden state variable representing the progress of the dialog so far, and a selected system action. The POMDP model defines two probabilistic dependencies: the conditional probability of the current state given the previous state and system action; and the conditional probability of the observation given the current state and previous system action.

A reward function specifies, for each turn, a fitness criterion as a function of the state and selected action for that turn. Given the reward function, it is possible to determine a policy that provides the optimal system action given what is known about the state distribution at the current time. This policy can then be used to generate system actions in the course of a dialog. Selecting system actions in order to maximize the reward is called planning.

To have a working system, the model parameters that define probabilities in the POMDP need to be estimated. This estimation is called learning. The parameters are typically estimated using a maximum likelihood (ML) criterion, rather than using the reward function. For example, a maximum likelihood dynamic Bayesian network (DBN) can be used. A major problem with those approaches is that planning and learning are optimized separately and independently using different criteria. In addition, learning and planning are notoriously difficult optimization problems because inference becomes intractable in variable spaces large enough to handle real problems.

SUMMARY OF THE INVENTION

The embodiments of the invention provide text and spoken dialog systems based on a statistical dialog framework. In contrast with a generative model used in conventional approaches, the invention uses a discriminative model to represent the relationship between system actions, observations, and other information based on a log-linear model framework. Then, the dialog manager outputs an appropriate system action given a sequence of previous observations and system actions by directly optimizing an expected reward using a belief propagation (BP) procedure.

Because the invention uses a log-linear model, various features obtained during dialogs can be incorporated in the model. The parameters in the log-linear model can be statistically trained by using dialog data based on the belief propagation procedure to improve performance using refined system actions.

The embodiments provide a coherent system that has the advantage of a consistent optimization criterion, and at the same time is more efficient to optimize. The dialog system is modeled using a log-linear probability distribution. Thus, the invention provides a log-linear dialog manager.

Log-linear distributions have been used to model sequences since the introduction of conditional random fields (CRF). Although log-linear models in general cannot represent all distribution families, their flexible use of feature functions enables the models to express a wide family of probabilistic models. Because the model is a Markov chain, efficient procedures can be exploited, for optimization. In particular, the embodiments optimize a sum of rewards along the time axis.

To represent the space of possible states, user actions, and system actions, context-free grammar (CFG) are used each of which is based on a graph of semantic representations related to the domain of the dialog system.

Instead of being simple multinomials, the random variables take values in the space of parse trees generated by the CFGs. This provides a rich structure that enables the extraction of a wide range of features. Because of the flexible use of features inherent in log-linear models, the features can be designed to make the dialog system behave exactly like a conventional rule-based dialog system as a special case. This is done by implementing the rules of the dialog system as indicator-function features, and initializing the parameters, such that the log-linear probability distributions correspond to these rules.

BRIEF DESCRIPTION OF THE FIGURES

FIG. 1 is a flow diagram of a planning part of a dialog manager according to embodiments of the invention;

FIG. 2 is a flow diagram of a learning part of a dialog manager according to embodiments of the invention;

FIG. 3 is an example parse tree according to embodiments of the invention;

FIG. 4 is a parse tree of an example state according to embodiments of the invention; and

FIG. 5 is a block diagram of example production rules for the parse tree of FIG. 4.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Planning

As shown in FIGS. 1 and 2, the embodiments of our invention provide a log-linear dialog manager. The system includes a planning part 100 and a learning part 200. The methods for the two parts can be performed in processors 150 and 250 connected to memory and input/output interfaces by busses as known in the art. Processors 150 and 250 can be combined.

System Model

Our probabilistic model has four variables at each time step. Two are observable variables: a system action a, 102 and an observation o, 101. The other two are latent variables are inferred: a user action u, 201 and the state s.
time t-1, the system prompts the user with a query \( a_{r-1} \). The response by the user is represented by \( o_r \). In one embodiment, \( o_r \) is a sequence of words spoken by the user. However, it is understood that response can be typed text, or the response can be entered into the system by other means.

The meaning of the response is represented by the user action \( u_r \), which can be inferred from the observation. The new state \( s_r \) can be inferred, based on the system action \( a_{r-1} \) and user action \( u_r \) and the previous state \( s_{r-1} \). In our system, the state \( s_r \) represents the user’s intention, although in general it could also include additional contextual information.

Using subscripted colons to denote sequences, e.g., \( s_{0:T} = \{ s_0, s_1, \ldots, s_T \} \), a dialog session of duration T is represented by four variable sequences: \( s_{0:T}, a_{0:T}, u_{0:T}, o_{0:T} \).

FIG. 3 shows the model for a dialog session as represented by a factor graph, which for our log-linear model corresponds to the following joint probability distribution over the variables:

\[
p(s_{0:T}, a_{0:T}, u_{0:T}, o_{0:T}) = \frac{1}{Z_o} \exp \left( \sum_{t=0}^{T} \theta_f \phi_f(s_t, a_t, u_t, o_{t+1}) + \sum_{t=1}^{T} \theta_g \phi_g(u_t, o_t) \right)
\]

where \( Z_o \) is a normalizing constant, \( \phi_f \) and \( \phi_g \) are vectors of feature functions, and \( \theta_f \) and \( \theta_g \) are vectors of the corresponding model parameters, respectively.

At time \( t=T \), \( s_{T+1} \) and \( u_{T+1} \) are undefined, so as shown in factor \( \theta_f \) of the factor graph. At time \( t=T \) we define \( \phi_f \) as a function of only its first two inputs. To simplify notation, we also define the following vectors:

\[
\theta = \begin{pmatrix} \theta_f \theta_g \end{pmatrix}, \quad \phi(t) = \begin{pmatrix} \phi_f(s_t, a_t, u_{t+1}) \\ \phi_g(u_t, o_t) \end{pmatrix}
\]

which enable us to rewrite equation (1) more succinctly as

\[
p(s_{0:T}, a_{0:T}, u_{0:T}, o_{0:T}) = \frac{1}{Z_o} \exp \left( \sum_{t=0}^{T} \theta^T \phi(t) \right)
\]

where \( Z_o = \exp \left( \sum_{t=0}^{T} \theta^T \phi(t) \right) \).

is the partition function of \( p(s_{0:T}, a_{0:T}, u_{0:T}, o_{0:T}) \).

Variable Spaces
We let S, U, A, and O represent the variable spaces, i.e., the set of all possible values for the variables \( s_r, u_r, a_r, \) and \( o_r \) respectively. Each observation \( o \in O \) can be waveforms, acoustic features, recognized texts, and/or linguistic features. We use \( o \) to represent the input sequence, and we define the variable space \( O \) as the set of all sequences of words in a vocabulary set \( V \).

We define each of the variable spaces S, U, and A using a context-free grammar (CFG) including a set of production rules. Each variable space is defined as the set of all possible parse trees that can be generated by its CFG.

FIG. 5 shows some of the production rules in the CFG that defines the variable space S. Each parse tree in S is a possible value of the state \( s_r \). FIG. 5 shows one possible value for state \( s_r \), which one parse tree in S that was generated using the production rules shown in boldface in FIG. 4. In FIG. 4, terminals are enclosed in quotation marks, while nonterminals are not. The variable SADDRESSSS can either be further extended with other production rules or remain as a free variable.

Features
As can be seen in the factor graph in FIG. 3 and in equation (1), there are two types of factors in our model. The first, denoted \( f \), models statistical dependencies between the previous and current state, the system action, and the user action. The second, denoted \( g \), models dependencies between observed word sequences and their semantic interpretations. For the variables whose spaces are defined using CFGs, we treat each variable value (each parse tree) as a set of active production rules. For example, the production rules that are active in the parse tree of FIG. 5 are shown in boldface in the production rules of FIG. 4.

Suppose \( G_x, G_y, \) and \( G_z \) are the set of production rules in the CFGs that define the variable spaces for S (states), U (user actions), and A (system actions), respectively. For factor \( g \), we associate each production rule in a user action with a language model for the associated word sequences. Specifically, given a user action \( u_r \) and observation \( o_r \), we have features of the form \( \phi_g(u_r, o_r) \), which indicates an indicator function that equals 1 if and only if a particular production rule \( k \in G_y \) is active in the parse tree of user action \( u_r \) and a particular bigram \( w_r \), \( w_{r+1} \) is present in the word sequence of observation \( o_r \).

The language model for a production rule that appears close to the root of the tree models a general class of utterance, whereas production rules that appear close to the leaves of the tree are more specialized. For factor \( f \), we can consider production rules that co-occur. For example, the feature \( \phi_f(s_{r-1}, k \in G_y) \) which concerns two particular production rules of \( k \), \( k' \in G_y \), equals 1 if and only if \( k \) is active in state \( s_{r-1} \) and \( k' \) is active in state \( s_r \). Another type of feature type typically seen in dialog systems is \( \phi_f(s_r, s_{r+1}, k \in G_y) \) which also requires that production rule \( j \in G_y \) is active in system action \( s_{r-1} \). This feature indicates that a particular system action tends to induce a particular state transition.

Planning and Learning
The two basic problems a dialog manager needs to solve are planning 100 and learning 200. We assume there is a reward function \( r: S \times A \rightarrow R^+ \) that assesses our model. We now describe the planning and learning in terms of the reward function.

Planning
Planning at time \( t \) is the problem of determining the optimal system action \( a_t \), given all previous system actions \( a_{0:t-1} \) and observations \( o_{0:t} \). Suppose the dialog has a duration \( T \). We define the planning problem as determining \( a_t \) to maximize the expected reward \( E \) as an objective function

\[
E = \frac{1}{T+1} \sum_{t=0}^{T} r(s_t, a_t \mid a_{0:t-1}, o_{0:t})
\]

The expectation is taken over all variables not given, i.e., all states, all user actions, and all future system actions and observations.

The objective function could be optimized exactly by hypothesizing each action \( a_t \), determining the expected reward given action that action using the sum-product procedure, and selecting the action that maximized expected reward.

However, for ease of implementation and speed, we instead optimize the objectives variational lower bound,
is the message from variable node $a_i$ to factor node $f_j$. We can use any distribution, including a uniform distribution where we do not assume any prior distributions for $a_i$.

$$a_i \sim \mathcal{U}(a_i)$$

is the message from variable node $s_i$ to factor node $f_j$.

$$f_j \sim \mathbf{m}(s_i)$$

is recursively determined from the previous step.

The message from variable node $u_{i+1}$ to factor node $f_i$ is determined from the distribution as

$$m_{u_{i+1} \leftarrow f_i} (a_{i+1}) = \sum_{s_{i+1}} \mathbf{m}(a_{i+1}) \exp \left[ \sum_{s_{i+1}} \mathbf{m}(s_{i+1}) \mathbf{m}(a_{i+1}) \right]$$

Thus, we avoid the summation over sequences

$$\sum_{a_{i+1}} \mathbf{m}(a_{i+1})$$

to determine the message

$$m_{u_{i+1} \leftarrow f_i} (u_{i+1})$$

The other messages can also be determined efficiently without computing the summation over the sequences based on the belief propagation methods.

Note that averaging over future actions using the sum-product procedure is different from conventional POMDP optimization, which seeks to maximize the reward over future system actions. It is also possible to use a max-product procedure on $a_i$ while using sum-product on the other variables to achieve maximization over future system actions. However, the model itself contains a stochastic policy that provides a predictive distribution over future actions.

Learning

The learning part 200 is similar to planning, except that instead of determining the optimal action we are interested in determining the optimal model parameters. In other words, we want to find $\theta$ such that the expected reward

$$R(\theta) = \mathbf{E}_{\gamma_{1:T}} \left[ \frac{1}{T+1} \sum_{t=0}^{T} r(s_t, a_t) \mid a_0, \gamma_1 \right]$$

Here,

$$m_{a_i \leftarrow f_i} (a_i)$$

is the message from variable node $a_i$ to factor node $f_i$. We can use any distribution, including a uniform distribution where we do not assume any prior distributions for $a_i$.

$$a_i \sim \mathcal{U}(a_i)$$

is the message from variable node $s_i$ to factor node $f_j$.

$$f_j \sim \mathbf{m}(s_i)$$

is recursively determined from the previous step.

The message from variable node $u_{i+1}$ to factor node $f_i$ is determined from the distribution as

$$m_{u_{i+1} \leftarrow f_i} (a_{i+1}) = \sum_{s_{i+1}} \mathbf{m}(a_{i+1}) \exp \left[ \sum_{s_{i+1}} \mathbf{m}(s_{i+1}) \mathbf{m}(a_{i+1}) \right]$$

Thus, we avoid the summation over sequences

$$\sum_{a_{i+1}} \mathbf{m}(a_{i+1})$$

to determine the message

$$m_{u_{i+1} \leftarrow f_i} (u_{i+1})$$

The other messages can also be determined efficiently without computing the summation over the sequences based on the belief propagation methods.

Note that averaging over future actions using the sum-product procedure is different from conventional POMDP optimization, which seeks to maximize the reward over future system actions. It is also possible to use a max-product procedure on $a_i$ while using sum-product on the other variables to achieve maximization over future system actions. However, the model itself contains a stochastic policy that provides a predictive distribution over future actions.

Learning

The learning part 200 is similar to planning, except that instead of determining the optimal action we are interested in determining the optimal model parameters. In other words, we want to find $\theta$ such that the expected reward

$$R(\theta) = \mathbf{E}_{\gamma_{1:T}} \left[ \frac{1}{T+1} \sum_{t=0}^{T} r(s_t, a_t) \mid a_0, \gamma_1 \right]$$

Here,

$$m_{a_i \leftarrow f_i} (a_i)$$

is the message from variable node $a_i$ to factor node $f_i$. We can use any distribution, including a uniform distribution where we do not assume any prior distributions for $a_i$.

$$a_i \sim \mathcal{U}(a_i)$$

is the message from variable node $s_i$ to factor node $f_j$.

$$f_j \sim \mathbf{m}(s_i)$$

is recursively determined from the previous step.

The message from variable node $u_{i+1}$ to factor node $f_i$ is determined from the distribution as

$$m_{u_{i+1} \leftarrow f_i} (a_{i+1}) = \sum_{s_{i+1}} \mathbf{m}(a_{i+1}) \exp \left[ \sum_{s_{i+1}} \mathbf{m}(s_{i+1}) \mathbf{m}(a_{i+1}) \right]$$

Thus, we avoid the summation over sequences

$$\sum_{a_{i+1}} \mathbf{m}(a_{i+1})$$

to determine the message

$$m_{u_{i+1} \leftarrow f_i} (u_{i+1})$$

The other messages can also be determined efficiently without computing the summation over the sequences based on the belief propagation methods.

Note that averaging over future actions using the sum-product procedure is different from conventional POMDP optimization, which seeks to maximize the reward over future system actions. It is also possible to use a max-product procedure on $a_i$ while using sum-product on the other variables to achieve maximization over future system actions. However, the model itself contains a stochastic policy that provides a predictive distribution over future actions.

Learning

The learning part 200 is similar to planning, except that instead of determining the optimal action we are interested in determining the optimal model parameters. In other words, we want to find $\theta$ such that the expected reward

$$R(\theta) = \mathbf{E}_{\gamma_{1:T}} \left[ \frac{1}{T+1} \sum_{t=0}^{T} r(s_t, a_t) \mid a_0, \gamma_1 \right]$$

Here,

$$m_{a_i \leftarrow f_i} (a_i)$$

is the message from variable node $a_i$ to factor node $f_i$. We can use any distribution, including a uniform distribution where we do not assume any prior distributions for $a_i$.

$$a_i \sim \mathcal{U}(a_i)$$

is the message from variable node $s_i$ to factor node $f_j$.

$$f_j \sim \mathbf{m}(s_i)$$

is recursively determined from the previous step.

The message from variable node $u_{i+1}$ to factor node $f_i$ is determined from the distribution as

$$m_{u_{i+1} \leftarrow f_i} (a_{i+1}) = \sum_{s_{i+1}} \mathbf{m}(a_{i+1}) \exp \left[ \sum_{s_{i+1}} \mathbf{m}(s_{i+1}) \mathbf{m}(a_{i+1}) \right]$$

Thus, we avoid the summation over sequences

$$\sum_{a_{i+1}} \mathbf{m}(a_{i+1})$$

to determine the message

$$m_{u_{i+1} \leftarrow f_i} (u_{i+1})$$

The other messages can also be determined efficiently without computing the summation over the sequences based on the belief propagation methods.

Note that averaging over future actions using the sum-product procedure is different from conventional POMDP optimization, which seeks to maximize the reward over future system actions. It is also possible to use a max-product procedure on $a_i$ while using sum-product on the other variables to achieve maximization over future system actions. However, the model itself contains a stochastic policy that provides a predictive distribution over future actions.

Learning

The learning part 200 is similar to planning, except that instead of determining the optimal action we are interested in determining the optimal model parameters. In other words, we want to find $\theta$ such that the expected reward

$$R(\theta) = \mathbf{E}_{\gamma_{1:T}} \left[ \frac{1}{T+1} \sum_{t=0}^{T} r(s_t, a_t) \mid a_0, \gamma_1 \right]$$

Here,
is maximized given all system actions \( a_{0:T} \) and all observations \( o_{1:T} \). Again the expectation is taken over all variables not given, namely all states and all user actions. Similar to the planning pass, we could also use the variational lower bound of equation (8) here.

We use gradient descent to optimize the learning objective. In general, for any utility function \( v(x) \) and probability distribution of the form based on the log-linear model

\[
p(x) = \frac{1}{Z_\theta} \exp(\theta^T \varphi(x)), \text{ where } Z_\theta = \sum_x \exp(\theta^T \varphi(x)),
\]

the derivative of the expected utility is:

\[
\frac{\partial}{\partial \theta} E_x[\text{v}(x)] = E_x[\varphi(x)v(x)] - E_x[\varphi(x)]E_x[\text{v}(x)].
\]

(10)

Note that for each parameter \( \theta_i \) in \( \theta \), the derivative is the covariance between the corresponding feature \( \varphi_i \) and the utility. Thus, the parameters corresponding to features that are positively correlated with utility are increased, while those whose corresponding features are negatively correlated with utility are decreased.

Applying this to our model gives:

\[
\frac{\partial \mathcal{R}(\theta)}{\partial \theta} =
\]

\[
\frac{\mathbb{E}_{m_{s_{1:T}}(s_{1:T})}}{m_{u_{1:T}}(u_{1:T})} \left[ \sum_{t=0}^{T} \varphi(t) \mathbb{E}_{m_{s_{1:T}}(s_{1:T})} \mathbb{E}_{m_{a_{1:T}}(a_{1:T})} \frac{r(s_t, a_t)}{T+1} \right] - \frac{\mathbb{E}_{m_{s_{1:T}}(s_{1:T})}}{m_{u_{1:T}}(u_{1:T})} \left[ \sum_{t=0}^{T} \varphi(t) \mathbb{E}_{m_{s_{1:T}}(s_{1:T})} \mathbb{E}_{m_{a_{1:T}}(a_{1:T})} \frac{r(s_t, a_t)}{T+1} \right],
\]

(11)

where expectations are determined using \( p(s_{0:T}, a_{1:T}, o_{1:T}) \). In the general case, it may be hard to determine these quantities. We use particle belief propagation.

Particle Belief Propagation

Because the variable spaces are too large to marginalize over, we solve the problem using particle belief propagation.

Consider a message \( m_{s_{1:T}}(s_{1:T}) \) passing from factor node \( f \) to variable node \( s_{1:T} \) by marginalizing over \( s, a, u \) and \( u_{1:T} \):

\[
m_{s_{1:T}}(s_{1:T}) =
\]

\[
\sum_{u_{1:T}} \left[ m_{u_{1:T}}(u_{1:T}) \cdot \mathbb{E}_{m_{u_{1:T}}(u_{1:T})} \mathbb{E}_{m_{s_{1:T}}(s_{1:T})} \frac{r(s_t, a_t)}{T+1} \right].
\]

If we rewrite the sum with importance sampling, then we obtain

\[
m_{s_{1:T}}(s_{1:T}) =
\]

\[
E_{u_{1:T}} \left[ m_{u_{1:T}}(u_{1:T}) \cdot \mathbb{E}_{m_{u_{1:T}}(u_{1:T})} \mathbb{E}_{m_{s_{1:T}}(s_{1:T})} \frac{r(s_t, a_t)}{T+1} \right].
\]

(12)

for some sampling distribution \( \pi_u(a), \pi_u(u), \pi_u(s) \) over which the expectation is determined.

We can then approximate the expectation with a sum

\[
m_{s_{1:T}}(s_{1:T}) = \frac{1}{N} \sum_{i=1}^{N} \exp(\theta^T \varphi(s_{1:T}, a_{1:T}, o_{1:T})) \cdot \frac{r_i(s_t, a_t)}{T+1},
\]

(13)

over samples \( (s^{(1)}_i, a^{(1)}_i, o^{(1)}_i), \ldots, (s^{(N)}_i, a^{(N)}_i, o^{(N)}_i) \).

Although the invention has been described by way of examples of preferred embodiments, it is to be understood that various other adaptations and modifications can be made within the spirit and scope of the invention. Therefore, it is the object of the appended claims to cover all such variations and modifications as come within the true spirit and scope of the invention.

1. A dialog manager comprising the steps of:
   receiving previous user actions and previous observations and current observations;
   hypothesizing previous and current user states, previous user actions, current user actions, future system actions, and future observations, wherein the user states, the user actions, and the user observations are hidden;
   extracting a feature vector based on the user states, the system actions, the user actions, and the observations;
   determining an expected reward of each current action based on a log-linear model using the feature vectors;
   and outputting the current action that has an optimal expected reward, wherein the steps are performed in a processor.

2. The dialog manager of claim 1, wherein a probabilistic model has four variables at each time step, including two observable variables: the system action \( a_t \), the observation \( o_t \), and two latent variables: the user action \( u_t \) and the user state \( s_t \).

3. The method of claim 2, wherein a dialog session of duration \( T \) is represented by four variable sequences \( s^T, a^T, o^T, u^T \).

4. The method of claim 3, wherein the dialog session is represented by a factor graph, which corresponds to a joint probability distribution

\[
\rho(s^T, a^T, o^T, u^T) =
\]

\[
\frac{1}{Z_\theta} \exp \left( \sum_{t=1}^{T} \theta^T \varphi(s_t, a_t, o_t, u_t) \right) + \sum_{t=1}^{T} \theta_n \varphi_n(a_t, o_t),
\]

where \( Z_\theta \) is a normalizing constant. \( \varphi \) and \( \varphi_n \) are the feature vectors, and \( \theta \) and \( \theta_n \) vectors of corresponding model parameters, respectively.

5. The method of claim 1, wherein the observations are spoken words or text.

6. The method of claim 3, wherein \( S, U, A, \) and \( O \) represent the variable spaces that are a set of all possible values for the variables \( s_t, u_t, a_t, \) and \( o_t \), respectively.

7. The method of claim 6, further comprising:
   defining the variable spaces \( S, U, \) and \( A \) using a context-free grammar (CFG) including a set of production rules.

8. The method of claim 7, wherein each variable space is defined as a set of all possible parse trees that can be generated by the CFG.
9. The method of claim 3, wherein a planning part of the dialog manager determines an optimal system action $a_t$, given all previous system actions $a_{t-1}$ and previous observations $o_{1:t}$.

10. The method of claim 3, further comprising:

maximizing an objective function

$$E_{\theta_1|\theta^{0:t}, o_{1:t}} \left[ \frac{1}{T+1} \sum_{t=0}^{T} r(o_t, a_t) \mid a_{t-1}, o_{1:t} \right]$$

to determine the expected reward.

11. The method of claim 10, further comprising:

optimizing a variational lower bound on the objective function.

12. The method of claim 10, wherein the objective function optimized using a gradient descent.

13. The method of claim 10, wherein the objective function is optimized using particle belief belief propagation.

* * * * *